REMARKS

This amendment is presented under the provisions of Rule 116(b) in response to the Final Office Action mailed on May 23, 2005, for the purpose of placing the application for allowance or in the alternative, in better form for Appeal. Claims 1-22 are active in the application.

In response to the Examiner's objection to the possible confusion arising from the use of the term blocking and unblocking in Applicants specification, Applicants have made minor amendments to paragraphs 42 and 105 which utilize such terms in the context of "buffering". Applicants have added the terms "buffered" and "into buffers" and "from buffers". This makes the use of such terms in conjunction with the term "records" consistent with other portions of the specification. The other use of the term "blocking" and "unblocking" in the context of "stopping execution" as noted by the Examiner in paragraph #502 should now be clear.

Also, Applicants have made minor changes to the claims. The amendments to the claims were made for the purpose of consistency in the steps set forth in the claims relative to "API" and for deleting redundant language in claims 5 and 15. Applicants believe that such amendments do not in any way raise new issues.

THE FINAL REJECTION SHOULD BE DEEMED PREMATURE

Relative to the present Office Action, Applicants find it most important to address the Examiner's response to Applicants arguments and in particular any new issues raised by such response. One such issue is the Examiner's objection to claims 1-22 as being unclear in the meaning of the terms "blocking" and "unblocking". Another issue is the Examiner's expanded use of personal knowledge in rejecting claims 5-7 and 15-17. These issues are discussed herein in conjunction with the basis of rejection of Applicants claims.

Claim Rejections under 35 USC § 102

Applicants traverse the Examiner's rejection of claims 1-3, 11-13, and 21-22 under 35 U.S.C. 102(b) as being anticipated by U.S Patent No. 6,714,968 to Prust (hereinafter Prust). Some of the arguments previously presented are reproduced herein for the convenience of the Examiner.

The Examiner Still Has Not Established a Prima Facie Case of Anticipation

The Examiner is required to show that each and every element as set forth in the claim is found in the Prust patent. Also, the Examiner must show that the identical invention is shown in as complete detail as contained in the claim and that the elements must be arranged as required by the claim.

Applicants traverse the Examiner statement that Prust teaches the invention as set forth in claim 1 directed to a method for accessing a first file on a disk system on one of a plurality of computer systems from a program executing on another of the plurality of computer systems utilizing first and second heterogeneous computer systems involving the operations of blocking a plurality of records into first plurality of blocks, transmitting the blocks during a first session from the 1st one of a plurality of computers to a 2nd one of the plurality of computers which unblocks the transmitted blocks into a 2nd plurality of records after closing the first session.

The Examiner has added as the basis of the original rejection that the technique for transferring data in between is by using FTP or browser and also cites Applicants' specification at paragraph #7, wherein FTP is characterized as a technique for transfer of files between heterogeneous computers in the prior art. Also, the Examiner cites paragraph #7 of Applicants specification relative to step (b) of Applicants' claim 1. From this, Applicants submit that the present rejection is in **essence a new rejection based** on the description pertaining to transferring data by FTP contained in paragraph #7 of Applicants specification. Applicants submit that this new rejection provides another reason as to why the final rejection should be deemed to be premature.

As discussed in the prior response, Prust is directed to a data storage system (e.g. storage servers) that provides **several methods of accessing virtual storage areas** by client computers. In one embodiment, access to the virtual storage areas is from a user directly via the operating system's user interface by calling standard file management routines provided by an API of the operating system (e.g. for copying files between hard disk and remote storage areas as well as renaming files and deleting files) (referred to herein as method 1). Prust states that in the method 1 embodiment, the client operating system of the client computers is the Macintosh operating system, such that the API includes the Apple File Services (AFS) and storage servers support accessing remote data

files within virtual storage areas via the Apple Filing Protocol (AFP) services over TCP/IP. Prust also discusses another embodiment of Figure 4 in which the operating system is a Windows operating system from Microsoft that incorporates the SMB protocol or WebDAV protocol. Here, it will be noted that the file services on the client and server systems are described by Prust **as being compatible or homogeneous**. Applicants direct the Examiner's attention to paragraphs #10-12 of Applicants specification which also discuss the use of functionality of writing files on remote systems by client computers prior art in **homogeneous systems** using the Sun Solaris operating system or Microsoft Windows operating system.

In another embodiment of Prust (herein cited as method 2), the user can access virtual storage areas by invoking a communications application such as a web browser or an FTP utility. In this embodiment, the communications application typically uses TCP/IP as the base protocol and additionally uses the HTTP protocol (used for transferring documents/text), the FTP protocol or even a proprietary data backup protocol. In this embodiment, the communications software application handles all communications with the storage servers and **the file management routines of the operating system (i.e. API) are not invoked**. Prust treats these two methods of access as being distinctly different.

In rejecting claim 1, the Examiner makes reference to portions of Prust that pertain to the above discussed two conceptually different embodiments of access methods #1 and#2. More specifically, relative to steps (A) and (B), the Examiner cites the embodiment of method #1 which uses the file management routines of the operating system API and relative to step (E), the Examiner cites the embodiment of method #2 (does not use the file management routines of the operating system (API)) and specifically cites the FTP protocol. Applicants submit that clearly, it is improper to combine the elements/steps of what Prust discloses as two different methods of accessing virtual storage areas in order to meet the requirements of 35 U.S.C. 102 as the Examiner has done. If this were not true, then one could pick and choose among different disclosures in a patent and combine them in any way that would arguably anticipate the claims in question.

Also, Applicants note that there are no specific portions of Prust or any other reference cited relative to steps (C) and (D) of claim 1. Hence, one would conclude that these steps are not anticipated and hence, the case for anticipation has not been shown. This is still another reason as to why the final rejection should be deemed premature.

As stated in Applicants previous amendment, Applicants find Prust absent first and second heterogeneous computer systems(discussed further herein), the use of an API in accessing of a first file in response to a program during a first session for the purpose of reading or writing plurality of records contained within such first file or the blocking of such plurality of records into blocks and the unblocking of such blocks into plurality of records by such heterogeneous computer systems in the manner set forth in claim 1.

Examiner's Response to Applicants Arguments in Claim 1 Rejection

The Examiner made several comments regarding Applicants arguments as to why claim 1 is patentable over the cited teachings of Prust. Applicants present the following rebuttals to these comments.

1. In response to the Examiner's statement that Applicants have not clearly defined the word "heterogeneous", Applicants submit that the term "heterogeneous" is to be accorded its commonly accepted meaning. Applicants specification discusses in the background section that the functionality that allows a computer to write files on another computer has been limited to homogeneous computers systems because there was no requirement to perform any translations between systems such as between 9 and 8 bit bytes as required in the preferred embodiment of the present invention (i.e. the first computer system is a GCOS 8 mainframe system that operates utilizing 36-bit words with either 4 9-bit or 6 6-bit characters per word and the preferred second computer system is a UNIX system utilizing 8-bit bytes). From this, Applicants submit that it should be clear from the specification that the term "heterogeneous" encompasses systems that have different operating systems that use different word structures. This definition is consistent with the prior art. For example, U.S. patent no. 5,758,125 defines ***Heterogeneous computer*** systems as computer systems with different storage architectures, computer systems with different central processing units (CPUs), computer systems with different disk organization methods, computer systems with different access methods, computer systems that use different controller interfaces etc. Similarly, U.S.

patent no. 6,289, 391 defines the term "heterogeneous multiprocessing system" to refer to a single computer system having two or more Central Processing Units (CPUs) that operate with a shared memory and utilize two or more different operating systems.

2. The Examiner states that since Prust clearly teaches using FTP or a browser as another embodiment for transferring files between the client and storage server and that since both FTP and browser are platform independent, they are inherently suitable for heterogeneous computers as admitted in paragraph #7 of Applicants specification. First of all, the embodiment (method #2) that uses FTP or a browser does not use the API file management routines of the operating system. But, it seems clear that both client and storage server would still need to use the same type of browser or FTP utilities which is the case in homogeneous systems as discussed in Applicants specification. Since Prust does not discuss the need for translating files as discussed in paragraph #7 of Applicants specification relative to heterogeneous systems, this is evidence that the Prust systems would be termed "homogeneous".

More specifically, as noted by the Examiner, paragraph #7 of Applicants specification does describe a prior art heterogeneous system shown in Figure 2. In that system, a first computer application writes records to a file. When the application completes writing to the file, the file is closed. Then, a utility such as FTP is utilized to transfer the file to a second computer system where a corresponding utility writes the file on disk on the second computer. A second application can read and process the second file. Paragraph #7 further states that **any necessary translations between the two heterogeneous computer systems are performed by the two utility programs**.

3. The Examiner states that the cited "packetizing data files" reads on the claims because each packet may be equivalent to a block. As discussed herein, the term "packetizes" refers to the process of converting the data files and metadata into "packets" for transfer over the network. As stated above, the description in Prust makes it clear that the user is accessing complete files and performing file management operations such as copying, renaming, moving and deleting files and directories (see column 6, lines 1-12). It should be obvious that if the files in question are sufficiently large relative to the size of a packet, then it would become necessary to break down the data in the files into a number of packets for transmission over the network. Thus, the process of "packetizing

data files" is opposite to the steps recited in Applicants claims. Packetizing is further discussed herein.

4. Additionally, the Examiner cites paragraph #42 of Applicants specification in support of the Examiner's opinion that Applicants admit that "data transfers between systems is typically on a (blocked) record basis" in prior art systems. A closer reading of paragraph #42 reveals that the paragraph pertains to the present invention and not to the prior art.

Paragraph #42 states the following: "Another **improvement** has been made to the prior art **(referring to the present invention)**. The application 130 on the first computer system 110 can specify what data conversions are to be performed by the interface between systems. Since the data transfers between systems is typically on a (blocked) record basis, this data conversion can be selected on a per field basis, and is performed on each selected field in each record transferred. Thus, some fields can be converted automatically from 36-bit integers to 32 bit integers (and potentially reversing the "endian" for the integers at the same time), while other fields can be converted from 9-bit ASCII to 8-bit ASCII." This latter aspect of the present invention is reflected in Applicants claims 9, 10 19 and 20. Also, in the prior art FTP method of moving data described in paragraph #7 which is discussed herein in the paragraph "Applicants Invention, an additional step would have been introduced; this step is labeled (3): (1) Reading data from a production database and writing it to a local file; (2) utilizing FTP to transfer the local file to a remote system; (3) **Running a process on the remote system that reads the transferred data, transforms it, and writes it to another file** and (4) Running a process on the remote system that uses the transferred data. From this discussion, it should be clear that paragraph #42 describes the present invention and not the prior art.

Applicants Invention

By contrast, the present invention enhances the prior art method of moving data between two systems. Consistent with the description in paragraph #7, the method of the prior art technique includes the basic steps of: (1) reading data from a data source such as a database and writing them to a local file with some possible restrictions on some data; (2) utilizing FTP to transfer the local file to a remote system; and (3) running a process

on the remote system that uses the transferred data. The present invention eliminates the need for step (2) and thus improves performance in addition to eliminating the need to integrate FTP into various operations such as for example, an unattended batch operation that can be performed overnight. At this point, it seems useful to discuss the prior art protocols cited by the Examiner in rejecting claim 1.

TCP/IP FTP

The Transmission Control Protocol and IP are layered protocols that fit below the network-applications protocols and above access and data-link protocols. The Internet Protocol resides at about the Network Layer (Layer 3) of OSI and TCP at the Transport Layer (Layer 4). File transfer systems making use of TCP/IP typically fall at Layer 5(Session). TCP/IP was developed as a means for tying together diverse networks with gateways. It uses checksums, sequence numbering of all data, retransmissions for reliability, reliable connection establishment and clearing and a flow-control mechanism.

The file-transfer process as it is managed by FTP is not intended to handle all issues or steps of the process just described. Rather, FTP presumes some basic properties, such as data type, file organization, and file ownership, and provides a means by which one computer can manipulate these properties on another computer system without either computer knowing details about the other. Files on a remote system are manipulated through a series of commands and responses, performing such functions as **get a file from** or **send a file to** a remote system. The File Transfer Protocol itself does not translate files from one computer type to another nor does it establish any kind of virtual network file. More fundamentally, it provides three dimensions: data types, file types and transmission modes. These dimensions can then be used by the two computers to establish a common ground. One of the transmission modes provided by FTP is **block mode that is usually used with very large files. In this mode, the source host breaks the data into well-defined blocks and the destination machine reassembles the blocks into an appropriate file.**

Problems such as data extraction, formatting and translation are accomplished prior or subsequent to submission to FTP in the Application Layer of the OSI standard. A file transfer using FTP can be initiated by either a human or another computer program. The human user will use a program called the **User FTP Program**. User FTP

provides access to all FTP services. In some installations, FTP may also be called (like a subroutine) from other application programs. This approach is used if there were complex data extraction, formatting and translation issues to be handled. (See pages 291-295 of the text entitled "Enterprise-Wide Computing: How to Implement and Manage LANs" by Thomas W. Madron, copyright © 1991 by Thomas W. Madron, published by John Wiley & Sons, Inc.) A copy of the cited material has been included for the convenience of the Examiner as Attachment A.

Additionally, U.S. patent no. 6, 718, 372 to Bober cited by the Examiner as being pertinent to Applicants disclosure also describes a prior art technique that uses FTP for obtaining access to data stored on a remote computer system in column 3 of the patent. The description in column 3 makes it clear that FTP transfers **the entire contents** of a requested file. According to Bober, the technique uses FTP to provide a connection between an FTP server and an FTP client **to transfer an entire file**, for example, from the mainframe to a workstation. According to the patent, a user application can invoke the FTP client directly using an FTP command to cause the FTP client to request **the entire contents** of one or more files from the FTP server. In response to such an FTP command, the FTP client provides standard FTP protocol messages over the network to the FTP server. In response to such messages, the FTP server finds and then transfers the **entire contents** of the requested file(s) obtained from the storage device back to the FTP client on the workstation via the network. The FTP client receives the data during the transfer and stores the data into a file created within the local storage device on the workstation. Once the transfer is complete, the FTP session is over and the application can access the copy of the requested file as needed directly on the local storage device.

Bober points out that the FTP is generally more limited in its capabilities than NFS **since FTP merely provides a complete local copy of an entire file.** Also, Bober does not consider FTP to be a true real time data access protocol in that the data access by an application takes place generally after the entire file has been transferred to the destination local storage device. Since FTP provides only a copy of the data file for use by the application, changes to the original file that occur after the FTP file transfer is complete may not be reflected in the copied version of the file stored within the local storage device.

From the above, it is clear that there are important distinctions between FTP file transfers and that of the present invention and from this it can be appreciated how performance can be improved by the removal of the FTP step by applying the teachings of the present invention. As set forth in claim 1, the present invention performs such moving of data by providing a record connection oriented API between the programs using it which in the preferred embodiment corresponds to GCOS 8 Cobol-85 API as described in Applicants specification (e.g. paragraphs #105, 332). The API is used by client programs in carrying out the steps of claim 1 as indicated. For example, step (A) of claim 1 would be carried out in the preferred embodiment by a program calling an **Open** function of the API that establishes a connection to the second computer (e.g. via the sockets interface which is at a lower layer of the OSI standard).

As indicated in step (B) of claim 1, the first computer blocks a first plurality of records into a first block of records via the API. In the preferred embodiment, the program calls a function **Write a Record** causing the Record Manager component to move a record into a collection buffer for the connection (see section 3.1.2 of and section 5.1.7 of Applicants specification). The first program repeatedly calls this Record Manager function resulting in the filling of the collection buffer. In the case of the present invention, since the blocking of records is done **prior to the call** to the sockets interface, there is no need to perform repeated communications with the sockets interface for each record (see section 4.1.3 of Applicants specification). This results in a substantial performance improvement. Applicants will now consider the basis of the Examiner's rejection of claim 1 relative to cited portions of the Prust patent.

Claim 1

For the reasons discussed above, the fact that Prust suggests that it may use a FTP protocol (method #2) should not be deemed equivalent of performing the steps in claim 1 of blocking a plurality of records, transmitting same, unblocking the blocks into a second plurality of records and closing the first session after completing the step of transmitting the blocks from a first one of the plurality of computer systems to a second one of the plurality of computer systems. As discussed above and in Prust, FTP is a networking protocol specifically for transporting the entire contents of **files** (not blocks of plurality of

records) from one computer on the network to another computer. Both embodiments (methods #1 and#2) envision moving the entire contents of files as discussed herein.

The material in lines 20-27 of column 5 cited by the Examiner relative to step (e) of claim 1 discusses that the operating system (method #1) **packetizes** data files and metadata received from the management routines and communicates the data to storage servers via a network. The term "packetizes" refers to the process of converting the data files and metadata into "packets" for transfer over the network. As stated above, the description in Prust makes it clear that the user is accessing complete files and performing file management operations such as copying, renaming, moving and deleting files and directories (see column 6, lines 1-12). These operations are like the typical operations a user would perform on complete files using "Windows Explorer" running on a personal computer.

Also, it should be obvious that if the files are sufficiently large relative to the size of a packet, then it would become necessary to break down the data in the files into a number of packets for transmission over the network. As discussed above, this takes place in a FTP transfer when transmitting in block mode. This process is the opposite of blocking data records into a first plurality of blocks and such process would occur at the session level and not at the application level as required in claim 1. Thus, the cited material indicates the use of packets that pertain to operations at the sockets interface level and not at the Application Layer of the OSI standard. Hence, Prust deals with data at the file level (complete files) and not with the elements of a specific file as specified in claim 1. Further, there is no mention of sessions in the cited material let alone closing a session after the occurrence of a specific event (completing the transmitting step (C).

Also, Applicants find that Prust is absent the use of an API or program for initiating access to a plurality of records within a file. Columns 1 and 6 of Prust discussed above mention an API but this is in the context of providing a user with access to file management operations of an operating system used for managing local files (lower level routines).

The absence of the above discussed elements and functions should be persuasive that Prust does not anticipate Applicants claim 1. A notice to this effect is respectfully solicited. If the Examiner persists in this rejection, Applicants ask that the Examiner to

be more specific as to the portions of Prust being relied on for concluding that claim 1 is anticipated.

Claim 2

The Examiner cites column 1, lines 49-67 discussed above that pertains to having a user accessing a virtual storage area using the file management API routines provided by the operating system and the retrieving data files from the storage area. Again, Prust is concerned with accessing complete files and not plurality of records. Also, as discussed above, Applicants find Prust absent the use of an API or program for initiating access to records within a file at the application level. Therefore, Applicants submit that Prust does not provide for the receiving of the first plurality of records via the API or the writing of the second plurality of records to the first file as set forth in claim 2. Accordingly, claim 2 should be deemed patentable over the cited portions of the Prust patent. A notice to this effect is respectfully solicited.

The Examiner commented relative to Applicants arguments pertaining to claim 2 that Prust reads on the claim because a file may constitute a plurality of records and therefore accessing a file is equivalent to accessing the plurality of records in its entirety. As discussed above, Prust views each file as a single entity in performing file management operations on such file and associating metadata with each data file to facilitate the quick cataloging and quick retrieval of the data file. Also, Prust discusses that a user in uploading files, attaches one or more data files. By contrast, claim 2 treats a plurality of records as individual entities which are collected to form a first plurality of blocks and are unblocked into a second plurality of records for writing into the first file. This is in contrast to writing a plurality of records as a first file using a single file-oriented command.

Claim 3

The Examiner cites column 1, lines 56-61 discussed above and column 7, lines 42-56. Column 7 discusses remote processing via centrally hosted applications and that a user may submit data files for remote processing by instructing the operating system 135 to copy the data files to a designated directory within the virtual storage area 225 by dragging and dropping the file onto the appropriate window presented by the operating system. This clearly describes a transfer of data files in a single direction. Applicants

find no discussion or support of the Examiner's conclusion pertaining to the storage server running an application program and transferring data back to the client computer in the discussed material pertaining to remote processing cited by the Examiner, let alone in the manner recited in claim 3. Accordingly, claim 3 should be deemed patentable over the cited portions of the Prust patent. A notice to this effect is respectfully solicited.

The Examiner comments relative to the arguments presented by Applicants that it is inherent that an FTP program is able to transfer files in both directions. In the cited material, the objective is to enable a user to access a virtual storage area on the remote system. Thus, Prust contemplates a different architecture from that suggested by the Examiner. Further, Applicants claim is not directed to an FTP transfer but rather to a method that eliminates the need for FTP transfers as discussed above.

Claims 11 and 13 and 21-22

In view of the above discussion, Applicants submit that claims 11 and 13 and 21-22 should also be deemed patentable for the same reason set forth relative to claims 1-3 and 11.

Claim Rejection - 35 U.S.C. § 103

Applicants submit that it is impermissible within the framework of section 103 to pick and choose from any one reference only so much of it as will support a given position, to the exclusion of other parts necessary to the full appreciation of what such reference fairly suggests to one of ordinary skill (see 147 USPQ at 393).

Claims 4 and 14

Applicants traverse the Examiner's rejection of claims 4 and 14 under 35 U.S.C. 103(a) as being unpatentable over Prust as applied to claims 1-3, 11-13 and 21-22 above further in view of Official Notice. The basis of the rejection is that (1) Prust fails to teach utilizing a credit based flow control mechanism to flow control the first plurality of blocks and utilizing a block based credit counting each of the first plurality of blocks as one credit, (2) Official Notice is taken that in a **fee for service business model** it is well known to **charge a client** according to the amount of data or traffic introduced by a service and (3) Official Notice is taken that utilizing a credit based flow control for limiting the number of transferred blocks is well known.

Applicants submit that the Examiner has not cited any reference defining the type of fee for service business model that is being officially noticed. Instead of providing any documentary evidence, the Examiner states what in the opinion of the Examiner is well-known in the art. Applicants submit that it is never appropriate to rely solely on "common knowledge" in the art without evidentiary support in the record, as the principal evidence upon which a rejection was based. (See Zurko, 258 F.3d at 1385, 59 USPQ2d at 1697 cited in Section 2144.03 of the MPEP).

Thus, in the absence of documentary evidence, it is not possible to understand fully the relationship between charging a fee for traffic that would be like charging a fee for making cell phone transmissions and a method of controlling the transmission of data as set forth in claims 4 and 14. The fact the Prust **may not provide** a free service, should not be deemed sufficient basis for concluding that certain limitations on the file transfer could have been established as stated by the Examiner. This is highly speculative.

Applicants' claims are directed to **data control** in the context of the proper **loading of data buffers and not to the charging of fees** as stated by the Examiner. As discussed herein, the user of the client computer in Prust that accesses a virtual storage area via a browser **or** FTP utility is required to make a request after each transmission that determines the flow of data. Therefore, Applicants find no motivation to utilize any flow mechanism in the Prust system, let alone the one defined in Applicants claims. In fact, Applicants submit that the addition of a fee service to place limitations on the amount of data transferred it would be contrary to the teachings of Prust. Moreover, the resulting combination assuming combination is possible would not accomplish the function performed by the credit based flow control mechanism of claims 4 and 14 (namely controlling the rate or flow of the plurality of blocks). Accordingly, Applicants submit that claims 4 and 14 should be deemed patentable over Prust and the undefined fee for service business model stated by the Examiner. A notice to this effect is respectfully solicited.

The Examiner comments that Applicants have challenged the legitimacy of using official notices rather than arguing the Examiner's ground of reasoning and requesting documentary evidence. Applicants submit that the above makes it clear that Applicants have in effect asked for documentary evidence stating that it is not possible to understand

fully the relationship between charging a fee for traffic and Applicants method of flow controlling the transmission of data. Applicants also cite the decision in Zurko which addresses the problems with relying solely on common knowledge. Applicants' position is that it is inappropriate in this case because of the problem in understanding how this knowledge teaches placing limitations on data transfers. Further, the Examiner has not addressed Applicants arguments why that such knowledge should not be deemed applicable to Prust. Applicants submit that limiting the number of blocks is not the equivalent of controlling the flow of blocks during transmission. For example, it will be appreciated that there is a significant difference between limiting the time spent on the telephone (by limiting the number of words spoken) and controlling the rate at which words are spoken during such time. In fact, the imposition of a fee would only discourage traffic just like the imposition of higher gas prices for discouraging car traffic.

Claims 5-8 and 15-18

Applicants traverse the Examiner's rejection of claims 5-8 and 15-18 under 35 U.S.C. 103(a) as being unpatentable over Prust as applied to claims 1-4, 11-14 and 21-22 above. Applicants submit for the reasons given above, claims 5-8 and 15-18 should be deemed patentable.

Claims 5-7

Additionally, as to claims 5-7, Applicants submit that the fact that the system cited by the Examiner as using a HTTP protocol as file transferring protocol makes it difficult to understand how this suggests the use of multiple sessions. It is well known that the HTTP protocol (hypertext transfer protocol) is one wherein an HTTP server is used to serve up **HTML documents** when requested by a client. It is further known that **the connection** between the client and server is usually **broken after the requested document or file has been transferred** (see Microsoft Computer Dictionary). Accordingly, in standard Internet browsing, connections are continually being made in response to client requests and broken before further transfers (upload or down load) can take place. That is, the current version of HTTP establishes a connection each time a client request is made. Therefore, Applicants find that it is not possible to conduct multiple TCP sessions as suggested by the Examiner, let alone in the manner set forth by claims 5-7. Further, Prust does not provide facilities or any suggestion of conducting

more than one session at a time. Other than the Examiner's suggestion of providing multiple sessions, Applicants find that there is no motivation for providing more than one session at a time. In view of the above, Applicants submit that claims 5-7 should be deemed patentable and a notice to this effect is respectfully solicited.

In response to the above, the Examiner comments that the Examiner cited an example that one may use a browser instance for uploading data to an internet service provider's allocated web site while at the same time browsing (down loading) the same site. In the original rejection of these claims, the Examiner did not make it clear that he was suggesting the use of multiple browser instances to establish multiple TCP sessions. Now, the Examiner is suggesting that a user merely open up multiple instances of a browser and establish connections between the client and multiple web sites. It has been noted (e.g. U.S. patent no. 6,101,482) that such an approach is quite impractical because of the significant amount of processing resources that would be consumed by the opening up and maintaining of multiple browser instances. Moreover, there would be no convenient way of maintaining the transaction information once the browser connections were terminated and/or using such information to complete some transaction.

At this point, it may be helpful to review the use of HTTP. In the Web environment which the Examiner cites, client machines effect transactions to Web servers use the Hypertext Transfer Protocol (HTTP), which is a known application protocol providing users access to files (e.g., text, graphics, images, sound, video, etc.) using a standard page description language known as Hypertext Markup Language (HTML). HTML provides basic document formatting and allows the developer to specify "links" to other servers and files. HTTP can be thought of as a lightweight file transfer protocol optimized for transferring **small files**. HTTP reduces the inefficiencies of the FTP protocol. HTTP runs on top of TCP/IP and was developed specifically for the transmission of hypertext between client and server. Additionally, it is well known that in the Internet paradigm, a network path to a server is identified by a so-called Uniform Resource Locator (URL) having a special syntax for defining a network connection. Use of an HTML-compatible browser (e.g., Netscape Navigator) at a client machine involves specification of a link via the URL. In response, the client makes a request to the server identified in the link and receives in return a document formatted according to HTML.

From the foregoing, it is seen that the internet or web environment requires the use of links for making connections and use of HTML documents. It seems that the Examiner may now be suggesting as grounds for rejection, the use of multiple browser instances for carrying out FTP transfers. If this is the case, it should be noted that Prust makes it clear relative to the embodiment of method #2 which does not invoke file management routines provided by the operating system API, the user can access the assigned virtual storage area by invoking a communications application **such as a web browser or an FTP utility.** Prust does not suggest invoking both applications. Further, as discussed above, Applicants have described how their invention provides an improvement over moving data by FTP transfers using an API.

Further, Prust envisions a different system which enables a user to access a virtual storage area allocated to the user for remote storage through the use of any necessary access information (password). This suggests the use of single password protected sessions rather that the use of multiple browser instances as proposed by the Examiner. Further, as discussed above, browser operations are directed to processing transactions involving text/documents and not to moving data records containing data for numerous transactions between multiple systems in response to a program as defined in claims 5-7.

Claim 8

The Examiner cites column 3, lines 23-40, column 9, lines 1-15 and column 8, lines 8-22 in addition to column 5, lines 34-37 and Figure 6. Column 3 discussed above indicates the use of different microprocessors for implementing computer 100. Column 9 contains claims 8-10 which are directed to the configuration of a web browser, a FTP utility, supporting WebDAV for accessing data files and an API that supports SMB protocol. Column 8 discusses the described seamless access to remote virtual data storage areas via a global computer network. Column 5 contains a similar discussion and Figure 6 illustrates a window 600 displayed by a conventional web browser when a user accesses a virtual storage area and lists each directory within the virtual storage area 225.

It is clear from the above that Prust uses standard browser/ internet/HTTP servers that may be interconnected. Further, Prust in column 3, lines 37-40 specify that client computer 100 represents any server, personal computer, laptop or even a battery powered pocket sized, mobile computer known as a hand held PC or PDA. . Further, the

background section of Prust discusses that prior art systems have limited modes for accessing data files such as for example, requiring a user to load proprietary software on their computers in order to communicate data files to the remote storage. Prust states in the summary of invention that his invention is directed to a system and method that can be configured to easily and seamlessly interact with a user's computer without **requiring proprietary software.**

Therefore, Applicants find that it would be contrary to the teachings of Prust and that there is no motivation to utilize mainframe proprietary computers, let alone the arrangement set forth in claim 8. Accordingly, claim 8 should be deemed patentable over the proposed combination suggested by the Examiner. A notice to this effect is respectfully solicited.

The Examiner has provided comments relative to Applicants arguments to the effect that the cited background section of Prust related to limited modes for accessing data files such as requiring a user to load proprietary software on their computers to communicate data files to remote storage would serve as motivation to design a system to overcome the difficulty. Further, the Examiner also comments that the prior art of record has shown its capability of transferring files over the internet for users of different platforms which obviously includes mainframe users. Applicants request that the Examiner be more specific in identifying which prior art is being relied on in support of his conclusion. In Prust, the transfers described involve HTML documents using HTTP or files by a FTP utility. If the Examiner is suggesting a browser interface as a front-end to a main frame computer, the result still would be a transfer of HTML documents using HTTP. Since Prust makes it clear that the disclosed system and method does not require proprietary software, it would be unlikely to provide communications between two different proprietary systems such as a mainframe computer system and UNIX based computer system as set forth in claim 8.

Claims 15-18

For the reasons given regarding claims 1, 5-8 and 11, claims 15-18 should also be deemed patentable. A notice to this effect is respectfully solicited.

Claims 9-10 and 19-20

Applicants traverse the Examiner's rejection of claims 9-10 and 19-20 under 35 U.S.C. 103(a) as being patentable over Prust as applied to claims 1-8, 11-18 and 21-22 above, further in view of Applicant admitted prior art (Figure 2 description).

Claims 9 and 10

Applicants' specification at page 2, lines 24-32 as noted by the Examiner, discusses Figure 2 as illustrating file reading and writing across heterogeneous systems in the prior art such as an FTP operation wherein any necessary translations between the two systems are performed by two utility programs. Applicants have shown how the claimed invention distinguishes over the system of Figure 2. Further, for the reasons discussed above, Applicants submit that Prust does not show or suggest the use of heterogeneous computers. The fact that Prust allows FTP transfers does not mean that the systems between which the transfers take place are heterogeneous. In fact, Applicants have shown that Prust uses compatible file handing systems eliminating the need for any data transformation.

Therefore, Applicants submit that there is no motivation to perform any data conversion operations let alone the type of conversion specified in claims 9 and 10. Accordingly, claims 9 and 10 should be deemed patentable and a notice to this effect is respectfully solicited.

The Examiner comments relative to Applicants arguments indicate that the Prust system is designed for the same operating system (suggestive of homogeneous systems) and for heterogeneous computers which enable one to access the virtual storage via platform independent tools such as browser and FTP utility. As discussed above, Prust states that either a web browser or an FTP utility can be used to access the virtual storage area. The cited description in column 6, line 45-column 7, line 34 discusses the web browser access relative to Figure 6 and access using a conventional electronic mail software application illustrated in Figure 7. It is well known to those skilled in the art that these types of operations do not involve data conversions. Moreover, these operations in Prust do not invoke the file management routines provided by an API of the operating system. This should be further evidence of any need to perform data conversion.

Claims 19 and 20

For the same reasons given relative to claims 1 and 9-10, Applicants submit that claims 19 and 20 should also be deemed patentable. A notice to this effect is respectfully solicited.

In view of the above arguments and amendment corrections, Applicants submit that claims 1-22 should be deemed patentable over the cited prior art. A notice to this effect is respectfully solicited. Additionally, Applicants have shown why the Final Rejection should be deemed premature. By withdrawing the Final Rejection, Applicants will have a fair opportunity to respond fully to the Examiner's claim rejections.

Applicants ask the Examiner to contact Applicants attorney upon receipt of this amendment for the purpose of advancing the prosecution of this application.

Further, if any questions or issues should arise with respect to this amendment or the allowability of this application, the Examiner is **urged** to **call Applicants' attorney at the number indicated herein.**

Respectfully submitted,

Faith F. Driscoll

Faith F. Driscoll
Registration No. 24,206
Attorney for Applicants
(781) 326-6645

Enc: Attachment A
FFD/fd

ATTACHMENT A

Enterprise-Wide Computing: How to Implement and Manage LANs

Thomas W. Madron

John Wiley & Sons, Inc.
New York ▪ Chichester ▪ Brisbane ▪ Toronto ▪ Singapore

Information coding. All information is coded in ASCII. Three ASCII control characters are used for message identification: SOH for numbered data messages, ENQ (Enquiry) for control messages (such as acknowledgements), and DLE (Data Link Escape) for maintenance messages. The remainder of the message, both text and header, is transparent.

Information transparency. The text or data field may be of any length from zero to two[14] (16,383) bytes of transparent data. Any ASCII character can be used for header and text. The header CRC is validated before COUNT is used to receive data.

Synchronization. Digital Data Communications Message Protocol provides for synchronized transmission and reception on byte and message levels. When synchronization occurs, two ASCII SYN characters are used preceding the SOH. Syncronization is not needed between messages as long as there are no gaps between them.

Communications facility transparency. Digital Data Communications Message Protocol is totally transparent and handles control character transparency efficiently. This means that DDCMP can be made to run over virtually any communications system regardless of the control facilities required by that system.

Higher Layer File-Transfer Mechanisms

The several protocols described in this section could, just as easily, have been incorporated into the previous section. They are separated because the previous section deals with the essentially proprietary aspects of IBM and DEC networking. This section, by way of contrast, deals with standardized or public-domain file-transfer techniques within broader networking architectures or techniques. These include TCP/IP's FTP, the OSI FTAM, and a brief description of EDI. The latter can, of course, be implemented within DEC or IBM networking architectures as well as within OSI or TCP/IP architectures.

TCP/IP FTP

Internetworking has become a major issue in most large organizations, although it is probably nowhere more complicated than in the DoD. Several years ago DoD's Defense Advanced Research Projects Agency (DARPA), through its Internet research program, developed standards for internetworking through its TCP/IP. Together they form a reliable, virtual-

circuit oriented, peer-to-peer communications protocol. Transmission Control Protocol and IP are layered protocols that fit below network-applications protocols and above access and data-link protocols. Internet Protocol resides at about the Network Layer (Layer 3) of OSI and TCP at the Transport Layer (Layer 4). File-transfer systems making use of TCP/IP typically fall at Layer 5 (Session).[12]

Transmission Control Protocol and Internet Protocol is not tied to any single type of network or technology. Rather, it was developed as a means for tying together diverse networks with gateways. The composite network provides the capability for devices on different physical networks to communicate as if they were on the same network. As a result of these objectives, TCP provides reliable, sequenced, byte-stream virtual circuits. It uses checksums, sequence numbering of all data, retransmissions for reliability, reliable connection establishment and clearing, and a flow-control mechanism. The features of TCP make it suitable for a variety of networks, making no assumptions about the reliability of the underlying network services. The TCP virtual circuit remains intact even when components of internetwork fail. Assuming that there are multiple paths between nodes, TCP will switch paths as necessary to guarantee connectivity. Internet Protocol provides TCP with addressing and internetworking capabilities. Addressing is based on a 32-bit address. Three formats for the addresses allow a range of network classes from small to large networks.[13]

The file-transfer process as it is managed by FTP is not intended to handle all the issues or steps of the process previously described. Rather, FTP presumes some basic properties, such as data type, file organization, and file ownership, and provides a means by which one computer can manipulate these properties on another computer system without either computer knowing details about the other. Files on a remote system are manipulated through a series of commands and responses, performing such functions as *get a file from*, or *send a file to* a remote system. File Transfer Protocol itself does not translate files from one computer type to another, nor does it establish any kind of virtual network file. More fundamentally, it provides three dimensions: data types, file types, and transmission modes. These dimensions can then be used by the two computers to establish a common ground. Problems such as data extraction, formatting, and translation (EBCDIC to ASCII, for example) are accomplished prior or subsequent to submission to FTP. A file transfer using FTP can be initiated by either a human or another computer program. The human user will use a program called the *User FTP Program*. User FTP provides access to all FTP

services. In some installations, FTP may also be called (like a subroutine) from other application programs. This would be the approach if there were complex data extraction, formatting, and translation issues to be handled.

File Transfer Protocol has been implemented on every size machine from large mainframes to personal computers under a variety of operating systems (OS) and environments. It rests, of course, on TCP/IP to actually get the data from one host to another. In the FTP process the initiating host constitutes a client and the remote host is the server. Files can be transferred in either direction, from client to server or server to client. Prior to file transfer, however, a remote login must be accomplished by the client, often using TELNET facilities. The client user must, of course, be authorized to access files on the server through whatever mechanisms are appropriate to the server. By providing a mechanism for negotiating a file transfer's options in three dimensions (data type, file type, and transfer mode), support has been provided for a variety of hardware and operating environments.

Data Types

Four data types are defined for FTP: ASCII, EBCDIC, Image, and Logical Byte Size. ASCII is the most common representation for FTP text files, although on IBM hosts such files would be EBCDIC. NVT ASCII consists of ASCII codes 0–127. The eighth bit (high-order bit) of the standard eight-bit byte is not used. When two IBM hosts are communicating with one another, it is often more convenient to use EBCDIC rather than translate from EBCDIC to ASCII and back. Both types may be further specified as to whether they will be presented to a line or page printer. Three options for providing printer control characters are offered: nonprint, TELNET formatting, and carriage-control formatting. With the latter, FORTRAN carriage-control conventions are used.

Image or binary data is used to exchange data between machines of the same type. No bits are skipped, preset, or otherwise modified. This data type is used to transfer executable programs or other data in binary form. Type Image could be used to transfer either ASCII or EBCDIC files since Image preserves all the information in the file; however, it would not be useful to use type Image to transfer files between different kinds of machines. Sometimes it is useful to store a binary file from one machine in a library on another, dissimilar, machine. For such purposes, the Logical Byte Size type is provided. Logical Byte Size takes an argument, which is

the size of the "byte." Size is somewhat arbitrary and in this instance does not conform to the normal definition of "byte."

File Types

Three file types are defined: File Structure, Record Structure, and Page Structure. For other file types, client and server computers must be programmed to take what is offered and place them in an appropriate format. File Transfer Protocol is designed to provide a very low level of file compatibility among computers and their OSs. Type File Structure is the simplest and most frequently used. It assumes that a file is simply a sequence of bytes (defined by the data type) followed by an End-of-File marker. The destination operating environment must make sense out of this approach. On an MS-DOS machine, for example, lines or records of a file are delimited by a carriage return/linefeed (<CRLF>) combination, and MS-DOS knows that when it sees <CRLF>, as do most applications running under MS-DOS. Some systems, however, prefer a defined Record Structure for files. Record Structure allows the transmission of individual records, separated by the standard End-of-Record marker for the specified data type. In the early days of ARPANET, when TCP/IP was being developed, DEC's TOPS-20 OS was prominent, and it used files composed of pages. The Page Structure of TOPS-20 continues its life in FTP, although TOPS-20 itself is outmoded and obsolete.

Transmission Modes

The Transmission Mode options are provided to optimize the use of the communications network. There are three transmission modes: Stream, Block, and Compressed. Stream Transmission is the default transmission mode for all transfers. In this mode the unmodified raw data are sent to TCP. This mode requires the lowest overhead since it imposes the lowest computational burden on the receiving machine. For Record Structure files a two-byte control code is used to mark the End-of-Record (EOR) and the End-of-File (EOF). The first byte is all ones, while the second byte has a value of one, two, or three to indicate EOR, EOF, or both. If a byte of all ones is to be sent, it is appended to a second byte of all ones. The second byte is stripped off by the receiving FTP. Block Mode is used to allow for restarting failed transfers and may be needed with very large files. In this mode, the source host breaks the data into well-defined blocks—the destination machine reassembles the blocks into an appropriate file. The client

and server FTPs cooperate to determine when blocks are successfully transferred and will request retransmission when this does not occur. The block mode is most like many of the other commonly used file-transfer mechanisms. Some files contain series of the same byte replicated many times. Since large files can sometimes take many hours to transfer, it is useful to compress files as much as possible before transmission. In compressed mode, the sending FTP removes excess bytes, while the receiving FTP expands the compressed data.

OSI FTAM

File Transfer Access and Management provides a set of services for transferring information between application processes and filestores. The specification is contained in ISO 8571, Parts 1–4 and requires, at least in the MAP implementation, ISO 8822 (Presentation Service Definition) and ISO 8823 (Presentation Protocol Specification). File Transfer Access and Management guarantees the ability to do the following:

- Work with binary or text files.
- Create files.
- Delete files.
- Transfer entire files
- Read file attributes.
- Change file attributes.
- Erase file contents.
- Locate specific records.
- Read and write records of a file.

The optionally supported file types include sequential files, random-access files, and single-key indexed sequential files.

A virtual filestore is defined by the FTAM standard, to describe the service provided by the FTAM service element. The function of the virtual filestore is to map FTAM services onto the local (real) filestore, thus making FTAM independent of any particular brand of computer. The FTAM standard does not provide an interface to the FTAM services, assuming that the implementor will define such an interface. The MAP specification, however, does define an FTAM application-interface specification, and those imple-